(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
22 February 2001 (22.02.2001)

PCT

(10) International Publication Number
WO 01/13202 A2

(51) International Patent Classification⁷: G06F 1/00

(21) International Application Number: PCT/US00/22675

(22) International Filing Date: 17 August 2000 (17.08.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/149,387 17 August 1999 (17.08.1999) US

(71) Applicant *(for all designated States except US)*: PROGRESS SOFTWARE, INC. [US/US]; 14 Oak Park, Bedford, MA 01730 (US).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: TUCKER, Richard [US/US]; 14 Oak Park, Bedford, MA 01730 (US).

(74) Agents: GREENBERG, Robert, A. et al.; Foley, Hoag & Eliot, LLP, One Post Office Square, Boston, MA 02109 (US).

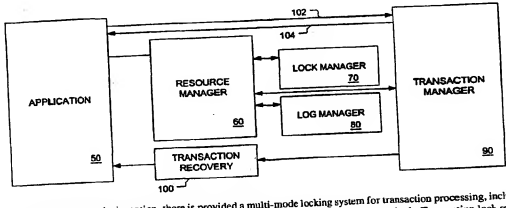
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *Without international search report and to be republished upon receipt of that report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: CONCURRENT COMMIT LOCK



(57) Abstract: According to the invention, there is provided a multi-mode locking system for transaction processing, including four modes of commit locking, such as a share lock, an update lock, a commit lock, and an exclusive lock. Transaction lock requests are queued according to a predetermined protocol, and may be upgraded under certain conditions. A system according to the invention may be used, for example, to perform a single-block logical operation without acquiring an update lock.

WO 01/13202 A2

CONCURRENT COMMIT LOCK

Reference to Related Applications:

This application claims priority from U.S. Provisional Application Serial No. 60/149,387, filed 08/17/99, entitled "CONCURRENT COMMIT LOCK".

Background of The Invention

1. Field of the Invention

This application relates to the field of transaction processing, and more particularly to the field of commit locks used in transaction processing.

2. Description of Related Art

Transaction processing has applications as diverse as telecommunications billing, stock trading, electronic mail, airline and hotel reservations, inventory planning, accounting, and factory process control. Transaction processing systems may have one or more databases, computer clients, a network, and application software to control operation of the system.

The properties of modern transaction processing systems include isolation and durability. Isolation dictates that each transaction appears to execute before or after each other transaction, i.e., that no two transactions are executed concurrently. Durability dictates that once a transaction is committed, that the effects are durable. If something goes wrong during execution of a transaction, a rollback operation is typically performed to "undo" the transaction. The capability to rollback transactions permits, for example, crash recovery of a transaction processing system and return to a previous, durable state. To implement these properties, transactions are generally bracketed by a begin-commit pair, with intermediate processing logged so that a transaction may be rolled back at

any point until it is committed. Transaction processing systems including these properties are described, for example, in "Transaction Processing: Concepts and Techniques", Jim Gray and Andreas Reuter, 1994.

Transaction management programs often perform physical logging for transaction "re-do" recovery and logical logging for transaction "undo." In physical logging, each change to a block in a database is separately logged. The information included in the log records is sufficient to "do" or "undo" on a block by block basis the changes logged. Logical operations, such as a logical operation to create a record in the database, are composed of one or more physical changes. Logical operations are undone by performing a compensating logical action, such as deleting the record from a database that a transaction created before rolling back.

Summary Of The Invention

According to the invention, there is provided a multi-mode locking system for transaction processing, including four modes of commit locking, such as a share lock, an update lock, a commit lock, and an exclusive lock. Transaction lock requests are queued according to a predetermined protocol, and may be upgraded under certain conditions. A system according to the invention may be used, for example, to perform a single-block logical operation without acquiring an update lock. The transaction processing program can use the lock in accordance with a prescribed protocol to ensure that only complete logical operations need to be undone while simultaneously allowing for highly concurrent processing of simultaneously executing transactions.

Brief Description Of Drawings

The foregoing and other objects and advantages of the invention will be appreciated more

fully from the following further description thereof, with reference to the accompanying drawings, wherein:

Fig. 1 is a block diagram of a transaction processing system that may be used with the invention;

Fig. 2 is a lock compatibility matrix according to the principles of the invention; and

Fig. 3 is a flow chart of a process for providing commit locks for transactions according to the principles of the invention.

Detailed Description of Certain Illustrated Embodiment(s)

Sometimes logical operations fail to complete. For example, in the middle of a "record create" operation, a system may crash or some other processing interruption may occur. Such incomplete logical operations raise an important question. For example, does a "record delete" operation have to be prepared to undo half a record create? This description details a concurrent commit locking method that determines when it is safe to begin logical undo (i.e., when the recovery log contains no incomplete logical operations).

To provide an overall understanding of the invention, certain illustrative embodiments will now be described, including a multi-mode commit lock having four locking modes. However, it will be understood by those of ordinary skill in the art that the methods and systems described herein can be suitably adapted to other transaction processing systems, and may have particular application in transaction processing systems that features physical logging with logical undo.

Figure 1 is a block diagram of a transaction processing system that may be used with the invention. A transaction processing system 10 may include an application 50, a resource manager

60, a lock manager 70, a log manager 80, a transaction manager 90, and transaction recovery functions 100. The components of the transaction processing system 10 generally cooperate to process transactions.

The application 50 may be any application or other process or program running on a computer. The application 50 may access other components of the transaction processing system locally, where the system 10 resides on a single computer, or through remote procedure calls where the system 10 is distributed. The application 50 may be, for example a client process in a client-server network. The components of the system 10 may communicate over a network based upon SNA, OSI, TCP/IP, DECnet, LAN Manager, or any other network protocol suitable for transmitting data among distributed processes. The transaction processing system 10 may be capable of maintaining communications with, and executing transactions for, any number of applications 50, whether local or remote, through a selection of suitable software and hardware.

The core services of the transaction processing system 10 include the resource manager 60, the lock manager 70, the log manager 80, and the transaction manager 90. The resource manager 60 may authorize, schedule and invoke services associated with a transaction. The resource manager 60 may also include any communications services and presentation services suitable to the application 50. The transaction manager 90 manages the commit and rollback of transactions, and recovery of transactions in the event of a failure. The log manager 80 may record a log of changes made by transactions to assist in rollback and reconstruction in case of failure. The lock manager 80 provides locking mechanisms to regulate concurrent access to objects stored by the resource manager 60, and may respond to lock requests from the resource manager 60 by granting, denying, or queuing requests for different types of locks on data, such as records stored in a relational database.

Transaction recovery functions 100 may be called following a failure, such as a system crash, to permit return to some definite state. The core services may be, for example, processes on a server that is accessible through remote procedure calls.

In one convention, the application 50 initiates a transaction with a Begin_Work() call to the transaction manager 90, as shown by a first arrow 102. The transaction manager 90 may respond with a transaction identifier that uniquely identifies the transaction, as shown by a second arrow 104. As a transaction proceeds with various work requests, these work requests may be forwarded from the application 50 to the resource manager 60, which coordinates database access, locking, logging, and so forth. The application 50 may conclude a transaction with a Commit_Work() call to the transaction manager 90, which may commit the transaction to a durable state.

The components of the transaction processing system 10 may be implemented on a variety of platforms. For example, records may be stored in any database system accessible by the resource manager 60, such as DB2, Rdb, Oracle, Informix, and Sybase. Each component of the transaction processing system may be implemented in a number of programming languages, such as COBOL, FORTRAN, PL/I, Ada, C, C++, Java, 4GL, and so forth. Further, the components of the transaction processing system 10 may be distributed over a number of servers in a networked environment, or may reside on a single server or a server cluster, such as those available from Compaq, Sun Microsystems, and IBM.

Figure 2 is a lock compatibility matrix according to the principles of the invention. In order to protect transaction histories for proper failure recovery, a lock should not be completed for any object, such as a database record, when that object is locked by another transaction in an

incompatible mode. The lock compatibility matrix 200 of Fig. 2 describes locking constraints for the compatibility of concurrent commit lock requests and lock modes for a multi-mode locking system. The lock compatibility matrix 200 may be used, for example, by the lock manager 80 of Fig. 1.

The mode of a request 202 is shown in a left-hand column in Fig. 2, and may include a "share", "update", "commit" and "exclusive" mode. The mode of a lock 204 is shown in a top row in Fig. 2, and may include a "share", "update", "commit" and "exclusive" mode. Where, for example, there is a request for a share lock 206, and the current lock for data is an update lock 208, the "Y" in the lock compatibility matrix 200 indicates that the request will be granted for the transaction. Where a lock mode is not granted, the request may be queued by the lock manager 80 until the lock mode is available for the record and/or transaction requested.

A share lock may be requested, for example, for any transaction or logical operation confined to a single block of data. A block of data is a fixed number of bytes such that when those bytes are written to non-volatile storage either all the bytes are written or none of them are. The share lock may be compatible with other share locks for other transactions. An update lock may be requested, for example, for any transaction or logical operation upon more than one block, such as a record in a database. The update lock may be compatible with other update locks, but incompatible with a commit lock. A commit lock may be requested, for example, when committing a transaction. The commit lock may be incompatible with update locks. The exclusive lock may be requested, for example, during back-up of a database, or any other operation requiring exclusive access to the database or a table or record therein.

A logical operation, as distinguished from a physical operation, may acquire an update mode

lock before modifying any database blocks. Once all database blocks that need to be modified to complete the logical operation have been modified, the lock may be released. Before committing a transaction, the database manager may acquire a commit mode lock. As may be seen in Fig. 2, if there are any update locks granted on the data, then the commit lock may not be granted. The lock request may be queued, and the transaction may not commit until the request is granted. Thus, according to the principles of the invention, transactions affecting single blocks of data, may be handled differently by the lock manager 80 than transactions affecting a record that spans multiple blocks.

Figure 3 is a flow chart of a process for providing locks for transactions according to the principles of the invention. The process 300 begins when a transaction is received by the transaction processing system, as shown in step 302. A transaction may be any function to be performed by the transaction processing system. The transaction manager may decompose a transaction into one or more logical operations that may be undone during crash recovery, which may be, for example, atomic database operations such as a row create, row delete, row modify, key add, key delete, and so forth.

As shown in step 304, each logical operation may be received by the resource manager. A lock may be requested for a logical operation from the lock manager, as shown in step 306. The type of lock requested may depend on the type of logical operation for which the lock is requested, including whether the operation acts upon a single block of data or on several blocks of date. It will further be appreciated that a lock request may be an upgrade request from a share lock to an update lock.

As shown in step 308, the lock manager determines whether a lock is available for the logical operation. This determination may be made by applying a lock compatibility matrix such as that shown in Fig. 2. If there is no lock on one or more blocks of data subject to a pending logical operation, then any requested lock may be granted. If a lock is available, then the process 300 may proceed to step 312 where a lock request is granted.

If a lock is not available in step 308, then the process 300 proceeds to step 310 where the lock request may be queued. Lock requests may be queued according to predetermined algorithms. For example, when an exclusive lock has been queued, all subsequent lock requests will be queued behind the exclusive lock. However, an upgrade request from share to update may be granted with a queued exclusive lock request. This may be useful where, for example, the upgrade is required to release locks on buffers for which an exclusive lock request is waiting. Share locks may be queued if there are any commit locks, and upgrade requests from share to update may be granted when there are pending commit lock requests. In this manner, lock starvation for a pending commit lock may be avoided as well as dead lock avoidance on database page locks.

In step 314, a logical operation may be performed. In step 316, the lock requested for the logical operation may be released upon completion of the logical operation.

In step 318, a determination is made of whether the transaction is complete. If the transaction is not complete, the process 300 may return to step 304 where a next logical operation of the transaction may be received. If the transaction is complete, then the process 300 may proceed to step 320 where a commit lock is requested. The request may be queued with other lock requests in a lock queue for prioritization and grant as described above. When the commit lock is granted 322, the

transaction may be committed 326, followed by the release 328 of the commit lock. After the commit 326, the transaction result is durable, and logging of the transaction by the log manager may be flushed as appropriate.

While the invention has been disclosed in connection with the preferred embodiments shown and described in detail, various modifications and improvements thereon will become readily apparent to those skilled in the art. Accordingly, the spirit and scope of the present invention is to be limited only by the following claims.

What is claimed is:

Claim(s)

1. A method for executing a transaction comprising:

providing a transaction to a memory;

requesting a lock for the transaction, the requested lock being a shared lock when the transaction affects a single block and the lock being an update lock when the transaction affects more than one block;

granting the lock request;

executing the transaction in the memory;

requesting a commit lock before committing the transaction; and

committing the transaction.

2. The method of claim 1 wherein the transaction comprises a plurality of logical operations.

3. The method of claim 1 further comprising requesting an exclusive lock to obtain exclusive control over data, and performing at least one of a backup or roll forward recovery log archival.

4. The method of claim 1 wherein the memory comprises a relational database.

5. The method of claim 1 wherein granting the lock request further comprises queuing the lock request and granting the requested lock when the requested lock

becomes available.

6. The method of claim 1, the requested lock further comprising an exclusive lock that provides exclusive control over modifying data.

7. The method of claim 1 wherein providing a transaction request further comprises providing a remote procedure call to a distributed transaction processing system.

8. Computer executable code for processing transactions comprising:

computer executable code that provides a transaction to a memory;

computer executable code that requests a lock for the transaction, the requested lock being a shared lock when the transaction affects a single block and the lock being an update lock when the transaction affects more than one block;

computer executable code that grants the lock request;

computer executable code that executes the transaction in the memory;

computer executable code that requests a commit lock before committing the transaction; and

computer executable code that commits the transaction.

9. The computer executable code of claim 8 wherein the transaction comprises a plurality of logical operations.

10. The computer executable code of claim 8 further comprising computer executable code that requests an exclusive lock to obtain exclusive control over data, and computer executable code that performs at least one of a backup or roll forward recovery log archival.

11. The computer executable code of claim 8 wherein the memory comprises a relational database.

12. The computer executable code of claim 8 wherein the computer executable code that grants the lock request further comprises computer executable code that queues the lock request and computer executable code that grants the requested lock when the requested lock becomes available.

13. The computer executable code of claim 8, the requested lock further comprising an exclusive lock that provides exclusive control over data.

14. The computer executable code of claim 8 wherein the computer executable code that provides a transaction request further comprises computer executable code that provides a remote procedure call to a distributed transaction processing system.

15. A multi-mode system lock for sequencing transactions in a transaction processing system, the the multi-mode system lock comprising:

computer executable code that provides a first lock mode, the first lock mode being requested for a transaction that operates on a single block of data;

computer executable code that provides an second lock mode, the second lock mode being requested for a transaction that operates on a plurality of blocks of data;

computer executable code that provides a third lock mode, the third lock mode being requested when a transaction is to be committed; and

computer executable code that provides a fourth lock mode, the fourth lock mode providing exclusive control of data.

16. The multi-mode system lock of claim 15, further comprising computer executable code for queuing a plurality of requests for locks.

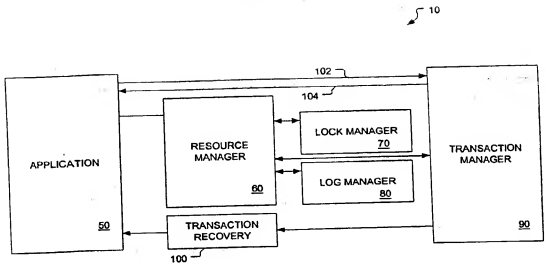
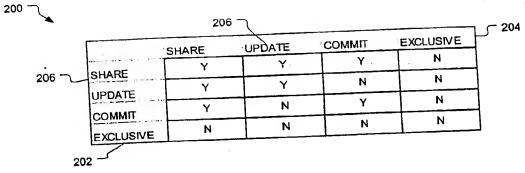

10
102
104
APPLICATION
50
RESOURCE MANAGER
60
LOCK MANAGER
70
LOG MANAGER
80
TRANSACTION MANAGER
90
TRANSACTION RECOVERY
100

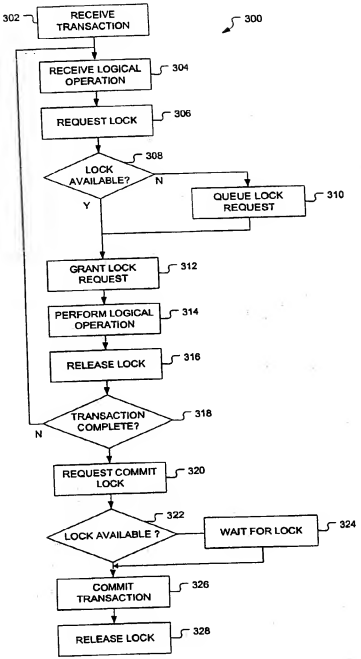
FIG. 1



	SHARE	UPDATE	COMMIT	EXCLUSIVE
SHARE	Y	Y	Y	N
UPDATE	Y	Y	N	N
COMMIT	Y	N	Y	N
EXCLUSIVE	N	N	N	N

FIG. 2

FIG. 3


300
302
RECEIVE TRANSACTION
RECEIVE LOGICAL OPERATION
304
REQUEST LOCK
306
LOCK AVAILABLE?
308
N
Y
QUEUE LOCK REQUEST
310
GRANT LOCK REQUEST
312
PERFORM LOGICAL OPERATION
314
RELEASE LOCK
316
TRANSACTION COMPLETE?
318
N
REQUEST COMMIT LOCK
320
LOCK AVAILABLE ?
322
WAIT FOR LOCK
324
COMMIT TRANSACTION
326
RELEASE LOCK
328

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
22 February 2001 (22.02.2001)

PCT

(10) International Publication Number
WO 01/13202 A3

(51) International Patent Classification[7]: G06F 9/46, 11/14

(21) International Application Number: PCT/US00/22675

(22) International Filing Date: 17 August 2000 (17.08.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/149,387 17 August 1999 (17.08.1999) US

(71) Applicant (*for all designated States except US*): PROGRESS SOFTWARE, INC. [US/US]; 14 Oak Park, Bedford, MA 01730 (US).

(72) Inventor; and
(75) Inventor/Applicant (*for US only*): TUCKER, Richard [US/US]; 14 Oak Park, Bedford, MA 01730 (US).

(74) Agents: GREENBERG, Robert, A. et al.; Foley, Hoag & Eliot, LLP, One Post Office Square, Boston, MA 02109 (US).

(81) Designated States (*national*): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

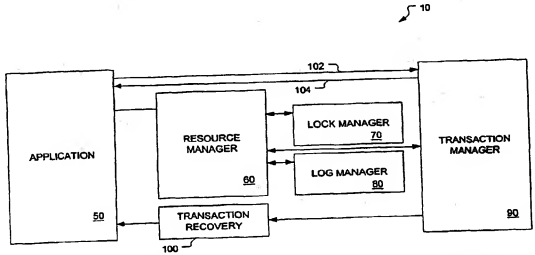
(84) Designated States (*regional*): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CI, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *With international search report.*

(88) Date of publication of the international search report: 3 May 2001

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: CONCURRENT COMMIT LOCK



(57) Abstract: According to the invention, there is provided a multi-mode locking system for transaction processing, including four modes of commit locking, such as a share lock, an update lock, a commit lock, and an exclusive lock. Transaction lock requests are queued according to a predetermined protocol, and may be upgraded under certain conditions. A system according to the invention may be used, for example, to perform a single-block logical operation without acquiring an update lock.

WO 01/13202 A3

INTERNATIONAL SEARCH REPORT

Int. ional Application No
PCT/US 00/22675

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F9/46 G06F11/14

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, WPI Data, PAJ

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	KWOK-YAN LAM: "AN IMPLEMENTATION FOR SMALL DATABASES WITH HIGH AVAILABILITY" OPERATING SYSTEMS REVIEW (SIGOPS),US,ACM HEADQUARTER. NEW YORK, vol. 25, no. 4, 1 October 1991 (1991-10-01), pages 77-87, XP000328788 page 79 -page 80, line 9 ---	1,8,15
A	DATE C J: "An Introduction to Database Systems (Volume 2)" ADDISON WESLEY, 1985, XP002159042 READING, USA page 83 -page 123, line 10 ---	1,8,15

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

* Special categories of cited documents :

A document defining the general state of the art which is not considered to be of particular relevance

E earlier document but published on or after the international filing date

L document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

O document referring to an oral disclosure, use, exhibition or other means

P document published prior to the international filing date but later than the priority date claimed

T later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

X document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

Y document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

& document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
31 January 2001	14/02/2001

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Absalom, R

2

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Int. ional Application No
PCT/US 00/22675

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	EP 0 100 821 A (INTERNATIONAL BUSINESS MACHINES CORPORATION) 22 February 1984 (1984-02-22) the whole document	1,8,15

2

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

information on patent family members

Intl Jonal Application No
PCT/US 00/22675

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 100821 A	22-02-1984	US 4648031 A	03-03-1987
		US 4507751 A	26-03-1985
		US 4509119 A	02-04-1985
		CA 1187190 A	14-05-1985
		DE 3379081 D	02-03-1989
		DE 3379353 D	13-04-1989
		DE 3380643 D	02-11-1989
		DE 3381324 D	19-04-1990
		DE 3381759 D	30-08-1990
		EP 0097234 A	04-01-1984
		EP 0097239 A	04-01-1984
		EP 0097256 A	04-01-1984
		EP 0097258 A	04-01-1984
		JP 1455877 C	25-08-1988
		JP 58225447 A	27-12-1983
		JP 63003341 B	22-01-1988
		EP 0096199 A	21-12-1983
		JP 58223857 A	26-12-1983
		DE 3379754 D	01-06-1989
		EP 0098928 A	25-01-1984
		JP 58223856 A	26-12-1983
		CA 1189979 A	02-07-1985
		JP 1380966 C	28-05-1987
		JP 59005483 A	12-01-1984
		JP 61050350 B	04-11-1986
		JP 59005361 A	12-01-1984
		CA 1191616 A	06-08-1985
		JP 59008072 A	17-01-1984